CONTAINER SHIPPING SYSTEM HLD

Container Shipping Management System

Document Information:

Field	Details
Document Version	V1.0
Created By	Patel Meet Bhaveshkumar
Date Created	04-08-2025
Project	Container Shipping Management Systen
Status	Approved

Table Of Contents:

---

1. Executive Summary

1.1 Project Overview

The Container Shipping Management System is a comprehensive digital platform designed to streamline global container shipping operations. The system manages end-to-end logistics including container tracking, route optimization, cargo management, and real-time visibility across the supply chain.

1.2 Key Objectives

- **Operational Efficiency**: Reduce manual processes by 70%
- **Real-time Visibility**: 100% container tracking capability
- **Cost Optimization**: 15% reduction in operational costs
- **Customer Experience**: Self-service portal with 24/7 access
- **Compliance**: Meet international shipping regulations

---

2. System Overview

2.1 Business Context

The global container shipping industry handles over 180 million TEU annually, with increasing demand for digital transformation and real-time visibility. Our system addresses critical pain points in traditional shipping operations.

2.2 Stakeholders

- **Primary Users**: Shipping operators, logistics managers, customers
- **Secondary Users**: Port authorities, customs officials, freight forwarders
- **System Administrators**: IT operations, security teams

2.3 System Scope

In Scope:

- Container lifecycle management
- Route planning and optimization
- Real-time tracking and monitoring
- Customer portal and notifications
- Billing and invoicing
- Reporting and analytics

Out of Scope:

- Physical container manufacturing
- Port infrastructure management

- Weather forecasting services
- Currency exchange calculations

---

3. Business Requirements

3.1 Functional Requirements

3.1.1 Container Management

- **REQ-001**: System shall track containers from origin to destination
- **REQ-002**: System shall manage container status (Available, In-Transit, At Port, Delivered)
- **REQ-003**: System shall support multiple container types (20ft, 40ft, High Cube, Refrigerated)

3.1.2 Route Management

- **REQ-004**: System shall calculate optimal routes based on cost, time, and capacity
- **REQ-005**: System shall support multi-modal transportation (Sea, Rail, Road)
- **REQ-006**: System shall handle route disruptions and re-routing

3.1.3 Customer Portal

- **REQ-007**: Customers shall view real-time container status
- **REQ-008**: Customers shall receive automated notifications
- **REQ-009**: Customers shall access shipping documents digitally

3.2 Non-Functional Requirements

3.2.1 Performance

- **NFR-001**: System response time < 2 seconds for 95% of requests
- **NFR-002**: Support 10,000 concurrent users
- **NFR-003**: Process 1 million transactions per day

3.2.2 Availability

- **NFR-004**: 99.9% system uptime (8.76 hours downtime/year)
- **NFR-005**: Maximum 30 seconds recovery time from failures

3.2.3 Security

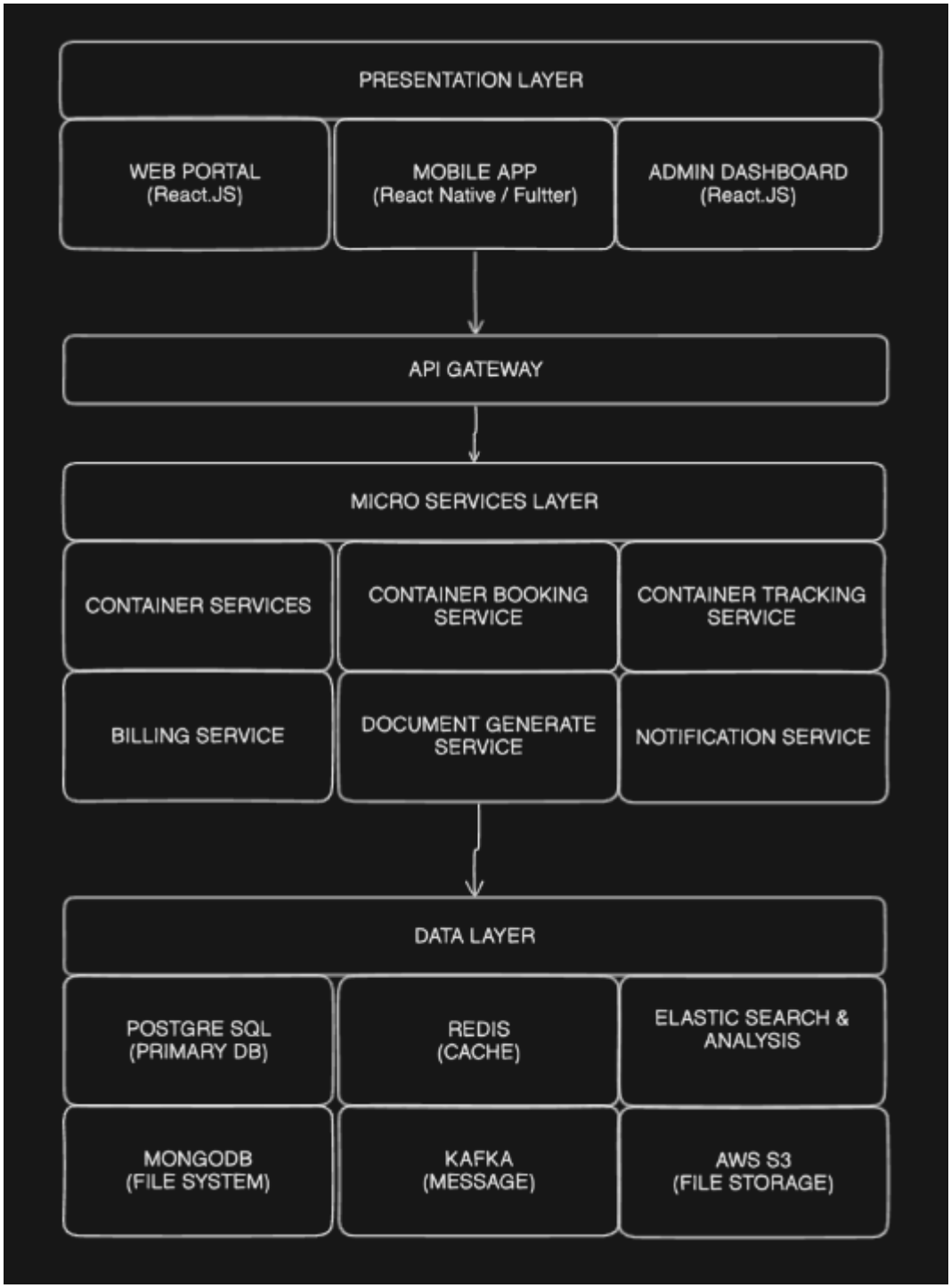
- **NFR-006**: All data encrypted in transit and at rest
- **NFR-007**: Multi-factor authentication for admin users
- **NFR-008**: Audit trail for all system activities

---

4. System Architecture:

4.1 Architecture Principles

- **Microservices**: Loosely coupled, independently deployable services
- **Event-Driven**: Asynchronous communication using message queues
- **Cloud-Native**: Containerized deployment with auto-scaling
- **API-First**: RESTful APIs for all system interactions



4.3 Architecture Patterns

- **Event Sourcing**: Track all changes as events
- **CQRS**: Separate read and write operations
- **Circuit Breaker**: Handle service failures gracefully
- **Bulkhead**: Isolate critical resources

---

5. Component Design

5.1 Core Services

5.1.1 Container Service

Purpose: Manage container lifecycle and tracking

Responsibilities:

- Container creation and assignment
- Status updates and tracking
- Capacity management
- Maintenance scheduling

Key APIs:

- POST /containers - Create new container
- GET /containers/{id} - Get container details
- PUT /containers/{id}/status - Update container status
- GET /containers/search - Search containers

Database Tables:

- containers - Container master data
- container_movements - Movement history
- container_status - Current status tracking

5.1.2 Route Service

Purpose: Optimize shipping routes and manage schedules

Responsibilities:

- Route calculation and optimization
- Schedule management
- Capacity planning
- Port coordination

Key APIs:

- POST /routes/calculate - Calculate optimal route
- GET /routes/{id} - Get route details
- PUT /routes/{id}/schedule - Update schedule
- GET /routes/vessel/{vesselId} - Get vessel routes

5.1.3 Customer Service

Purpose: Manage customer information and interactions

Responsibilities:

- Customer registration and profile management
- Authentication and authorization
- Booking management
- Communication preferences

---

6. Database Design:

6.1 Database Strategy

- PostgreSQL: Primary transactional data
- MongoDB: Document storage (contracts, manifests)
- Redis: Caching and session management
- Elasticsearch: Search and analytics

6.2 Core Entity Relationships

```
CREATE TABLE containers (
    container_id UUID PRIMARY KEY,
    container_number VARCHAR(20) UNIQUE NOT NULL,
    container_type_id INT REFERENCES container_types(id),
    size_type ENUM('20ft', '40ft', '45ft'),
    status ENUM('Available', 'In-Use', 'Maintenance', 'Retired'),
    current_location_id UUID REFERENCES locations(id),
    owner_id UUID REFERENCES customers(id),
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE shipments (
    shipment_id UUID PRIMARY KEY,
    shipment_number VARCHAR(30) UNIQUE NOT NULL,
    customer_id UUID REFERENCES customers(id),
    origin_port_id UUID REFERENCES ports(id),
    destination_port_id UUID REFERENCES ports(id),
    estimated_departure TIMESTAMP,
    estimated_arrival TIMESTAMP,
    actual_departure TIMESTAMP,
    actual_arrival TIMESTAMP,
    status ENUM('Booked', 'In-Transit', 'Delivered', 'Cancelled'),
    created_at TIMESTAMP DEFAULT NOW()
);
CREATE TABLE routes (
    route_id UUID PRIMARY KEY,
    route_name VARCHAR(100),
    origin_port_id UUID REFERENCES ports(id),
    destination_port_id UUID REFERENCES ports(id),
    distance_km DECIMAL(10,2),
    estimated_duration_hours INT,
    active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT NOW()
);
```

6.3 Data Archival Strategy

- **Hot Data**: Last 6 months (Primary DB)
- **Warm Data**: 6 months - 2 years (Compressed storage)
- **Cold Data**: >2 years (Archive storage)

---

7. API Design

7.1 API Standards

- **REST**: Follows RESTful principles
- **JSON**: Standard data format

- **HTTP Status Codes**: Proper status code usage
- **Versioning**: URL-based versioning (/v1/, /v2/)

7.2 Core API Endpoint:

7.2.1 Container Management API:

```
# Get Container Details
GET /api/v1/containers/{containerId}
Response:
{
  "containerId": "CTR-001",
  "containerNumber": "CSLU3054383",
  "type": "40ft",
  "status": "In-Transit",
  "currentLocation": {
    "latitude": 25.276987,
    "longitude": 55.296249,
    "port": "Dubai Port"
  },
  "route": {
    "origin": "Shanghai Port",
    "destination": "Los Angeles Port",
    "estimatedArrival": "2024-08-15T10:00:00Z"
  }
}

# Update Container Status
PUT /api/v1/containers/{containerId}/status
Request:
{
  "status": "At-Port",
  "location": {
    "portId": "PORT-LA-001",
    "timestamp": "2024-08-10T14:30:00Z"
```

```
    },
    "notes": "Container arrived at LA Port"
  }
```

7.2.2 Tracking API

```
  # Track Container
  GET /api/v1/tracking/{containerNumber}
  Response:
  {
    "containerNumber": "CSLU3054383",
    "currentStatus": "In-Transit",
    "timeline": [
      {
        "status": "Loaded",
        "location": "Shanghai Port",
        "timestamp": "2024-07-20T08:00:00Z"
      },
      {
        "status": "Departed",
        "location": "Shanghai Port",
        "timestamp": "2024-07-21T14:00:00Z"
      }
    ],
    "estimatedArrival": "2024-08-15T10:00:00Z"
  }
```

7.3 Error Handling:

```
{
  "error": {
    "code": "CONTAINER_NOT_FOUND",
    "message": "Container with ID CTR-001 not found",
    "details": {
      "requestId": "req-12345",
      "timestamp": "2024-08-04T10:30:00Z"
```

```
    }
  }
}
```

8. Security Architecture

8.1.1 Network Security

- **Firewall**: Cloud-based WAF (Web Application Firewall)
- **DDoS Protection**: Rate limiting and traffic analysis
- **VPN**: Secure admin access
- **Network Segmentation**: Isolated network zones

8.1.2 Application Security

- **Authentication**: OAuth 2.0 + JWT tokens
- **Authorization**: Role-based access control (RBAC)
- **Input Validation**: SQL injection and XSS prevention
- **Encryption**: AES-256 for data at rest, TLS 1.3 for transit

8.1.3 Data Security

- **Database Encryption**: Transparent data encryption
- **Key Management**: AWS KMS / Azure Key Vault
- **Data Masking**: PII data protection in non-prod environments
- **Backup Encryption**: Encrypted backups with retention policy

8.2 Authentication Flow:

User → Login Request → Auth Service → Validate Credentials

↓

JWT Token Generated ← Auth Service ← User Verified

↓

API Requests with Token → API Gateway → Token Validation

↓

Access Granted → Microservice → Response to User

8.3 Compliance

- **GDPR**: Data privacy and right to be forgotten
- **SOC 2**: Security and availability controls
- **ISO 27001**: Information security management

- **Maritime Security**: ISPS Code compliance

---

9. Scalability & Performance

9.1 Scalability Strategy

9.1.1 Horizontal Scaling

- **Microservices**: Independent scaling of services
- **Load Balancers**: Distribute traffic across instances
- **Auto-scaling**: Dynamic resource allocation
- **Database Sharding**: Distribute data across multiple databases

9.1.2 Caching Strategy

- **Application Cache**: Redis for session and frequently accessed data
- **Database Cache**: Query result caching
- **CDN**: Static content delivery
- **Browser Cache**: Client-side caching for assets

9.2 Performance Optimization

9.2.1 Database Optimization

- **Indexing**: Optimized indexes for query performance
- **Connection Pooling**: Efficient database connections
- **Read Replicas**: Separate read and write operations
- **Query Optimization**: Regular query performance analysis

9.2.2 Application Optimization

- **Lazy Loading**: Load data on demand
- **Pagination**: Limit result set sizes
- **Compression**: Gzip compression for API responses
- **Async Processing**: Background job processing

9.3 Performance Metrics:

Metric	Target	Monitoring
API Response Time	< 2 seconds	Real-time
Database Query Time	< 500ms	Real-time
Page Load Time	< 3 seconds	Synthetic tests
Throughput	1000 req/sec	Load testing

10. Technology Stack:

10.1 Frontend Technologies:

Component	Technology	Justification
Web Portal	React.JS	Mordern, Comment – based with strong ecosystem
Mobile app	React Native	Cross platform
Admin Dashboard	React.JS + Material UI	Rich component libarary, consistent api
State management	Redux Toolkit	Predictable state management

10.2 Database Technologies:

Component	Technology	Justification
Primary Database	PostgreSQL 14	ACID Compilance, JSON support, reliable
Document Store	MongoDb	Flexible Scheme, document storage.
Cache	Redies 7.0	In-memory performance, data structures
Search Engine	Elasticsearch 8.0	Full-text search, analytics

10.3 Backend Technologies:

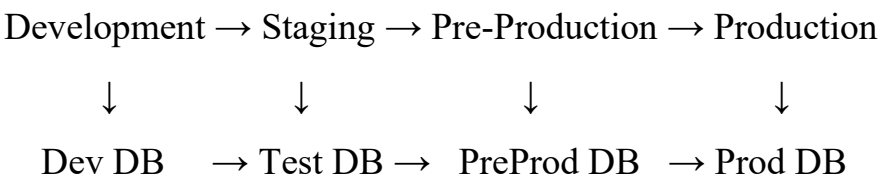
Component	Technology	Justification
API Gateway	Axios	High Performance, plugin ecosystem
Microservices	Node.js + Express	JavaScript ecosystem, async processing
Authentication	Auth0	Managed service, OAuth compliance
Message Queue	Kafka	Reliable Message Delivery, clustering

10.4 Infrastructure:

Component	Technology	Justification
Cloud Provider	AWS	Global Presence, maritime industry adoption
Containerization	Docker + Kubernetes	Scalability, orchestration
CI/CD	GitLab CI/CD	Integrated pipeline, container registry
Monitoring	Prometheus + Grafana	Open Source, Comprehensive metrics

11. Deployment Architecture:

11.1 Environment Strategy:



11.2 Production Deployment:

```
# Kubernetes Deployment Configuration
apiVersion: apps/v1
kind: Deployment
metadata:
  name: container-service
spec:
  replicas: 3
  selector:
    matchLabels:
      app: container-service
  template:
    metadata:
      labels:
        app: container-service
    spec:
      containers:
      - name: container-service
        image: shipping/container-service:v1.0
        ports:
        - containerPort: 3000
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: db-secret
```

```
        key: database-url
    resources:
      requests:
        memory: "256Mi"
        cpu: "250m"
      limits:
        memory: "512Mi"
        cpu: "500m"
```

11.3 Disaster Recovery

- RTO: Recovery Time Objective < 4 hours
- RPO: Recovery Point Objective < 1 hour
- Backup Strategy: Daily automated backups with 30-day retention
- Multi-Region: Active-passive deployment across regions

---

12. Monitoring & logging

12.1 Monitoring Stack:

- Application Metrics: Prometheus + Grafana
- Infrastructure Metrics: CloudWatch / Azure Monitor
- Log Aggregation: ELK Stack (Elasticsearch, Logstash, Kibana)
- APM: New Relic / Datadog for application performance

12.2 Key Metrics Dashboard:

```
Business Metrics:
├── Total Containers Tracked: 45,230
├── Active Shipments: 1,247
├── On-Time Delivery Rate: 94.5%
└── Customer Satisfaction: 4.3/5

Technical Metrics:
├── System Uptime: 99.97%
├── Average Response Time: 1.2s
├── Error Rate: 0.03%
└── Active Users: 2,341
```

12.3 Alerting Strategy

Alert Level	Response Time	Escalation
Critical	Immediate	On-call engineer + Manager
Warning	15 minutes	On-call engineer
Info	1 hour	Email notification

---

13. Risk Assessment:

13.1 Technical Risks:

Risk	Impact	Portability	Mitigation
Database Failue	High	Low	Master solve application, automated
API Gateway Outage	High	Medium	Multiple gateway instances, health checks
Third-party Service Dependency	Medium	Medium	Circuit breakers, fallback mechanisms
Data Breach	High	Low	Encryption, access controls, security audits

13.2 Business Risks:

Risk	Impact	Portability	Mitigation
Regulatory Changes	Medium	High	Compliance monitoring, flexible architecture
Market Competition	High	High	Continuous innovation, customer feedback
Economic Downturn	High	Medium	Cost optimization, scalable infrastructure

13.3 Operational Risks

Risk	Impact	Portability	Mitigation
Key Personnel Loss	Medium	Medium	Compliance monitoring, flexible architecture
Vendor Lock-in	Medium	Low	Multi-cloud strategy, open standards
Capacity Planning	Medium	Medium	Auto-scaling, performance monitoring

14. Timeline & Milestones:

14.1 Project Phases:

Phase 1: Foundation (Months 1-3)

├── Core microservices development

├── Database setup and basic APIs

├── Authentication system

└── Basic UI components

Phase 2: Core Features (Months 4-6)

├── Container tracking functionality

├── Route optimization

├── Customer portal

└── Mobile app MVP

Phase 3: Advanced Features (Months 7-9)

├── Real-time notifications

├── Analytics and reporting

├── Document management

└── Billing system

Phase 4: Production & Scale (Months 10-12)

├── Performance optimization

├── Security hardening

├── Production deployment

└── User training and adoption

14.2 Key Milestones

Milestone	Target Date	Success Criteria
MVP Release	Month 6	Core tracking features functional
Beta Launch	Month 9	100 beta customers onboarded
Production Go-Live	Month 12	Full system deployment
Scale Achievement	Month 15	10,000+ containers tracked

15. Appendices

15.1 Glossary

- **TEU**: Twenty-foot Equivalent Unit
- **Bill of Lading**: Legal document between shipper and carrier
- **Port of Loading (POL)**: Origin port
- **Port of Discharge (POD)**: Destination port
- **ETA**: Estimated Time of Arrival
- **ETD**: Estimated Time of Departure

15.2 Assumptions

- Global internet connectivity available at all ports
- Third-party APIs (weather, port schedules) remain available
- Container IoT sensors provide reliable data
- Regulatory requirements remain stable during development

15.3 Dependencies

- **External APIs**: Port schedules, weather services, customs systems
- **Hardware**: IoT sensors, GPS tracking devices
- **Third-party Services**: Payment gateways, notification services
- **Regulatory Approvals**: Maritime authority compliance

15.4 References

- Maritime Industry Standards
- Container Shipping Best Practices
- International Maritime Organization Guidelines
- Port Community System Standards